Effiziente sequentielle und parallele Verfahren zur Erkennung von Schlüsselbildern und zum Retrieval von digitalen Videos

Stefan Geisler

Institut für Informatik, Technische Universität Clausthal,
Julius-Albert-Strasse 4, 38678 Clausthal-Zellerfeld
Stefan.Geisler@informatik.tu-clausthal.de

Zusammenfassung

In der vorliegenden Arbeit werden Verfahren zur Suche von Bildern und Objekten in Videos untersucht. Der Schwerpunkt liegt auf der Effizienz der Verfahren, die es zum einen erlauben, in Echtzeit während einer Videoübertragung Schlüsselbilder zu erkennen, um beispielsweise eine Aufnahme zu starten, und zum zweiten eine hohe Anzahl Videos, wie sie in einer Videodatenbank gespeichert sind, zu durchsuchen. Für den zweiten Anwendungsfall werden parallele Verfahren entwickelt und auf verschiedenen Parallelrechnern vergleichend untersucht.

1 Einleitung

Die Anzahl digitaler Videos nimmt täglich in enormem Maße zu. In Fernsehstudios hat die digitale Technik in den letzten Jahren Einzug gehalten, Redakteure arbeiten genauso mit digitalem Material wie auch die eigenen Sendungen und auch das Programm anderer Sender digital archiviert wird. In Privathaushalten ist seit dem Erfolg der DVD – und in ihrem Schatten der VCD, SVCD oder neuerdings DivX/MPEG-4 Videodatenträgern – der Umgang mit digitalen Videos ebenfalls alltäglich. Verstärkt wird diese Tendenz durch digitale Fernsehstandards (DVB-T, DVB-S und DVB-C [1]) sowie digitale Videorekorder.

Als Videocodec kommen im Wesentlichen die Standards der MOVING PICTURE EXPERTS GROUP MPEG-1 [2], das technisch ähnliche MPEG-2 [3], sowie die aktuelle Fortentwicklung MPEG-4 [4] zum Einsatz. Durch die offenen Standards ist eine recht hohe Interoperabilität der einzelnen Systeme gegeben. Als Alternativen zu MPEG-4 sind proprietäre Systeme von MICROSOFT (WINDOWS MEDIA 9 [5]) und von REALNETWORKS (REALMEDIA [6]) auf dem Markt vertreten, jedoch im Unterhaltungsbereich noch von geringer Bedeutung.

Mit der steigenden Anzahl digitaler Videos wird auch der Bedarf an Verfahren und Methoden zur Speicherung und zum effizienten Wiederfinden von Videos größer. Seit

Jahren wird daher an der Entwicklung von Videodatenbanken gearbeitet. Die prominentesten Vertreter in der Forschungslandschaft sind VIDEOQ [7], die VIRAGE VIDEO ENGINE [8] und CUEVIDEO [9].

Ein Anspruch an *echte* Videodatenbanken im Vergleich zu Datenbanken, die *auch* Videos speichern können, ist die Arbeit direkt auf dem Videomaterial und nicht nur die Verwaltung von Stichwortlisten zu jeder Videodatei. Ziel ist es, den aufwändigen Prozess der Erstellung von Schlüsselwörtern zu umgehen. Bereits wenige Sekunden eines Videos erfordern mehrere Minuten Arbeit einer geschulten Person, um die Beschreibung anzufertigen. Dieser hohe Aufwand bringt sowohl eine Verzögerung, die insbesondere im Nachrichtensektor problematisch werden kann, als auch nicht zu vernachlässigende Kosten mit sich. Ferner bleibt selbst bei festgelegten Stichwortlisten die Subjektivität der Beschreibung ein Problem, da das Bildmaterial in verschiedenem Kontext unterschiedliche Bedeutung haben kann.

Neben der Archivierung von Videos ist weiterhin die automatische Auswahl interessanter Ausschnitte aus einer laufenden Übertragung von Interesse. Der Benutzer spart auf diese Weise nicht nur enorme Mengen an Speicherplatz, sondern darüber hinaus auch noch die Zeit, die er ansonsten benötigen würde, das wichtige vom unwichtigen Material zu trennen. Ein populäres Beispiel ist das Überspringen von Werbung, das bereits in die Consumer-Elektronik Einzug gehalten hat.

In diesem Artikel sollen beide Aspekte betrachtet werden. Dazu wurde ein Prototyp für das MPEG-1-Format entwickelt, dessen Resultate auf das MPEG-2-Format übertragbar sind. Der Artikel ist wie folgt gegliedert: Nachdem im nächsten Abschnitt die dynamische Suche erläutert wurde, werden in Kapitel 3 die wesentlichen Eigenschaften der Videokodierung des verwendeten Formats beschrieben. Diese bilden die Grundlage für die im sich daran anschließenden Abschnitt vorgestellten Optimierungen zur effizienten Suche in Videos. In Kapitel 5 werden die verwendeten Methoden zur Messung der Bildähnlichkeit eingeführt. Im darauf folgenden Kapitel wird die Qualität der dadurch erzielten Ergebnisse diskutiert. Daran schließen sich zwei Kapitel an, in denen parallele Algorithmen und Architekturen zur Lösung der Aufgabe vorgestellt werden und ihre Leistungsfähigkeit für die betrachteten Anwendungsfälle diskutiert wird. Den Abschluss bilden Fazit und Ausblick.

2 Dynamische Suche in Videodatenbanken

Während es bei der Erkennung gesuchter Bilder in laufenden Übertragungen, etwa ein charakteristisches Bild zu Beginn einer Sendung, wodurch die Aufnahme gestartet werden soll, auf der Hand liegt, dass bestimmte zum Vergleich benötigte Merkmale im Video nach der Anfrage berechnet werden, ist dies bei Datenbanken nicht gleich offensichtlich.

Bei Videodatenbanken können Merkmale wie Farbe, Form oder Bewegung beim Einfügen eines neuen Videos in die Datenbank berechnet werden. Nach der Anfrageübermittlung brauchen diese Merkmale nur aus der Datenbank ausgelesen und nicht aufwändig berechnet werden. Diese a-priori-Berechnung ist jedoch nur so lange an-

wendbar, wie lediglich nach ganzen Bildern gesucht wird und nicht nach Bildausschnitten, wie etwa Personen oder Objekten. KAO beschreibt für Bilddatenbanken daher einen Ansatz für dynamisches Retrieval [10, 11].

Demnach übermittelt der Benutzer ein Anfragebild oder notfalls auch eine selbst angefertigte Skizze an das System mit dem Auftrag, eine Menge von Bildern zurückzuliefern, die dem Anfragebild ähnlich sind oder die Bildbereiche beinhalten, die dem Anfragebild ähneln. Zu diesem Zweck wird das Suchbild über jedes Bild der Datenbank gelegt und schrittweise an verschiedene Positionen verschoben (Template Matching). An jeder Position wird die Differenz zu dem darunter liegenden Bereich errechnet und die Bilder mit dem geringsten Unterschied bilden die Ergebnismenge. So gelingt es, dieselbe Person vor unterschiedlichem Hintergrund zu finden. Neben dem direkten Vergleich ist ferner der Vergleich mit einem skalierten oder rotierten Template möglich, was zu einer höheren Genauigkeit führt. Der Nachteil der dynamischen Suche ist ein hoher Rechenaufwand nach der Übermittlung der Anfrage, dem derzeit nur durch Parallelisierung begegnet werden kann, um zu für den Benutzer akzeptablen Antwortzeiten zu gelangen.

Die im Folgenden betrachtete Suche in einer Videodatenbank arbeitet nach dem gleichen Grundprinzip. Die Anforderungen an die Rechenleistung steigen mit dem Wechsel von Bildern zu Videos weiter an. Daher wird für den Fall der Videodatenbanken untersucht, wie spezielle Eigenschaften der MPEG-Videos zur Beschleunigung der Suche ausgenutzt werden können.

3 Die Videoformate MPEG-1 und MPEG-2

Die Standards zur Videokodierung werden jeweils im zweiten Teil der Norm [2, 3] festgelegt. Sie beruhen grundsätzlich auf einer sehr ähnlichen Vorgehensweise, wobei im MPEG-2-Format eine höhere Bildqualität bei gleichzeitig höherer Bitrate erzielt wird.

Das MPEG-1 und das MPEG-2-Verfahren teilen das Video in einzelne Bilder (*frames*) auf, die inhaltlich nicht weiter unterteilt werden. Je nach Format und gewünschter Auflösung findet eine Unterabtastung statt, wobei die Helligkeit zumeist in höherer Auflösung als die Farbkomponenten des verwendeten YC_bC_r-Farbsystems gespeichert werden. Vier verschiedene Bildtypen werden dabei unterschieden:

- I-Frame: In einem *intra-coded picture* werden sämtliche Informationen des Bildes gespeichert. Es kann ohne Kenntnis anderer Bilder dekodiert und dargestellt werden.

- P-Frame: Die *predictive coded pictures* können Verweise auf zeitlich zurückliegende Bilder enthalten. Einzelne Bildteile brauchen so nicht erneut gespeichert zu werden, wenn ein ähnlicher Bildteil in einem vorherigen Bild gefunden wurde. Stattdessen werden nur die Differenzen gespeichert, was erheblich weniger Speicherplatz benötigt, jedoch anfälliger für Übertragungsfehler ist.

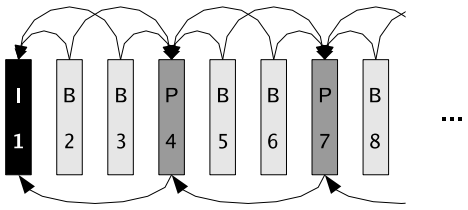

Abbildung 1: Typische Abfolge der Frametypen im MPEG-1/2-Format in der Darstellungsreihenfolge.

- B-Frame: Die Speicherung von *bidirectionally predictive pictures* ist ähnlich zu der der P-Frames, nur mit dem Unterschied, dass Verweise sowohl auf ein vorheriges als auch auf ein zukünftiges Bild möglich sind.

- D-Frame: *DC-coded pictures* existieren nur im MPEG-1-Format und haben auch dort nahezu nie Anwendung gefunden. Sie speichern das Bild in grober Auflösung, um die Implementierung eines schnellen Vor- und Rücklaufs zu erleichtern.

Abbildung 1 zeigt eine typische Bildfolge und die darin enthaltenen Verweise.

Jedes Bild wird in Blöcke der Größe 8×8 unterteilt, wobei vier Helligkeitsblöcke und jeweils ein korrespondierender Farbblock zu einem Makroblock zusammengefasst werden. Intra-kodierte Blöcke werden der Diskreten Cosinustransformation (DCT) unterworfen, an die sich eine Quantisierung anschließt. Dadurch werden viele der Koeffizienten, insbesondere bei den vom menschlichen Auge nur schlecht wahrgenommenen hohen Frequenzen, zu Null. Die Koeffizienten können im letzten Schritt mit einer Lauflängen- und anschließenden Huffmankodierung Platz sparend gespeichert werden.

4 Effiziente Suche in Videos

Für eine effiziente Bildsuche in digitalen Videos ist es von hoher Bedeutung, die Anzahl der zu vergleichenden Bilder ebenso gering zu halten, wie den Aufwand für die Dekodierung. Im entwickelten Prototyp wurden daher zwei Schritte zur Effizienzsteigerung eingeführt, die im Folgenden vorgestellt werden sollen.

4.1 Reduzierung der Bildanzahl

Die Suche in jedem einzelnen Bild eines PAL-Video mit 25 fps würde eine zu hohe Rechenleistung erfordern. Sie kann jedoch wegen der meist hohen Ähnlichkeit benachbarter Frames auf etwa zwei Bilder pro Sekunde beschränkt werden. Ein anderer denkbarer Ansatz wäre der Vergleich des Suchbildes mit einem Schlüsselbild für jede Szene. Dabei besteht jedoch die Gefahr, dass das gesuchte Objekt nicht auf dem Schlüsselbild

zu erkennen ist, z.B. weil es nur kurz in der Szene auftaucht oder zeitweise von einem anderen Objekt verdeckt wird.

Die meisten MPEG-Encoder erzeugen Videobitstreams mit einer Framefolge, in der etwa jedes 12. oder 15. Bild intrakodiert ist. Die Suche kann demnach genau auf die Menge der I-Frames eingeschränkt werden. Hiermit ist ein wesentlicher Vorteil verbunden: Das Dekodieren von I-Frames ist vergleichsweise einfach, weil keine Referenzen zu anderen Bildern aufgelöst werden müssen und somit P- und B-Frames komplett übersprungen werden können.

4.2 Arbeiten auf komprimiertem Videomaterial

Wie bereits beschrieben, sind die einzelnen Bilder in Blöcke der Größe 8×8 Pixel aufgeteilt, auf die die DCT angewandt wird. In der komprimierten Datei werden nur die Koeffizienten der DCT gespeichert. Da die inverse DCT der rechenzeitaufwändigste Schritt bei der Dekodierung ist, wird die Suche nicht auf den dekodierten Pixeln durchgeführt, sondern auf den Koeffizienten der DCT. Zu diesem Zweck muss das Vergleichsbild in gleicher Weise transformiert werden, was jedoch nur ein einziges Mal zu erfolgen hat.

Weiterhin wird die Geschwindigkeit dadurch erhöht, dass nur jeweils der erste Koeffizient eines Blockes verglichen wird. Der erste Koeffizient, auch DC-Koeffizient genannt, repräsentiert den Gleichanteil oder anders gesagt den Durchschnittswert des Helligkeits- bzw. Farbblocks. Das entspricht einem Vergleich der Bilder bei einer um den Faktor 64 reduzierten Auflösung.

Ein ähnlicher Ansatz für JPEG-Bilder findet sich bei CHANG et al.[13]. FALKEMEIER et al. [14] sowie SHEN et al. [15] arbeiten zur Gliederung und Indizierung von Videos ebenfalls auf komprimiertem Material.

5 Methoden zur Messung der Ähnlichkeit von Bildern

Eine Vielzahl von Funktionen zur Messung der Ähnlichkeit zweier Bilder ist in den letzten Jahren veröffentlicht worden. Hier sollen nur die Verfahren kurz vorgestellt werden, die später auch auf die Suche in Videos angewandt werden.

5.1 Pixelweiser Vergleich

Das naheliegendste Verfahren ist der pixelweise Vergleich der beiden Bilder P und Q, deren Ähnlichkeit gemessen werden soll. Hierbei kommt üblicherweise die euklidische Abstandsnorm zur Anwendung:

$$d_{pixel}(P,Q) = \left(\sum_{i=1}^{N} \sum_{c=1}^{C} (p_{ci} - q_{ci})^2 \right)^{\frac{1}{2}}$$

Dabei seien N die Anzahl der Pixel im Bild, C die Anzahl der Farbkanäle und p_{ci} der Pixel an Position i des Farbkanals c im Bild P.

Das Ergebnis ist stark ortsabhängig, was sich je nach Anwendungsgebiet sowohl als Vor- wie auch als Nachteil erweisen kann.

5.2 Farbmomente

Das Merkmal der Farbmomente wurde 1995 von Stricker und Orengo eingeführt [12]. Jedes Bild wird dabei als Wahrscheinlichkeitsverteilung aufgefasst, die durch ihre Momente beschrieben werden kann.

Das erste Moment ist der Erwartungswert μ. Im Allgemeinen ist das k-te Moment der diskreten Zufallsvariablen X wie folgt definiert:

$$E(X^k) = \sum_i x_i^k P(X = x_i)$$

Die Zentralmomente werden dann folgendermaßen definiert:

$$E((X - \mu)^k) = \sum_i (x_i - \mu)^k P(X = x_i)$$

Für die Farbmomente werden daraus abgeleitet für jeden Farbkanal c die drei folgenden Werte berechnet:

$$\mu_c(P) = \frac{1}{N} \sum_{i=1}^{N} p_{ci} \ ,$$

$$\sigma_c(P) = (\frac{1}{N} \sum_{i=1}^{N} (p_{ci} - \mu_c)^2)^{\frac{1}{2}} \ \text{ und}$$

$$\gamma_c(P) = (\frac{1}{N} \sum_{i=1}^{N} (p_{ci} - \mu_c)^3)^{\frac{1}{3}}.$$

Die Ähnlichkeit zweier Bilder P und Q wird dann wie folgt definiert, wobei die Gewichte w_{c1} bis w_{c3} je nach verwendetem Farbraum und Vorwissen über die Bilder angepasst werden können:

$$d_{mom}(P, Q) = \sum_{c=1}^{C} w_{c1} |\mu_c(P) - \mu_c(Q)| + w_{c2} |\sigma_c(P) - \sigma_c(Q)| + w_{c3} |\gamma_c(P) - \gamma_c(Q)|$$

Da die ersten drei Momente die Verteilung nicht eindeutig beschreiben, kann d_{mom} auch für nicht gleiche Bilder Null werden. Aus diesem Grund handelt es sich nicht um eine Metrik, sondern lediglich um eine Ähnlichkeitsfunktion. In der praktischen Anwendung haben sich die Farbmomente als sehr robust erwiesen.

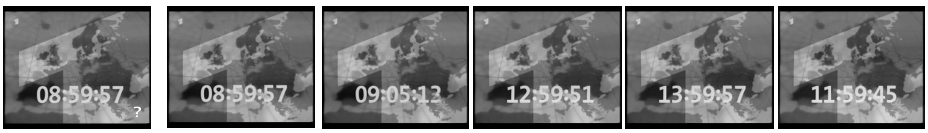

Abbildung 2: Der Vorspann der Tagesschau kann sowohl mit dem Vergleich der DC-Koeffizeinten als auch der Farbmomente sicher mittels eines Beispielbildes erkannt werden.

5.3 Weitere Methoden

Alternative Methoden basieren auf Histogrammen, Formen, Texturen oder auch auf dem Vergleich von Wavelet-Koeffizienten. Mit Ausnahme der Histogramm-Verfahren erfordern diese Methoden einen erheblich höheren Rechenaufwand als die vorgestellten Verfahren und werden daher in dieser Arbeit nicht betrachtet. Sie können jedoch leicht ergänzt werden und würden von den meisten der vorgenommenen und im Folgenden erläuterten Optimierungsschritte automatisch profitieren. Eine Übersicht von Verfahren zum Vergleich von Bildern findet sich in den Kapiteln 7 und 8 in [10].

6 Qualität der Suchergebnisse

Zur Bewertung der Qualität der Suchverfahren wurde eine Testmenge von über 15 Stunden Videomaterial erzeugt. Das entspricht 1,35 Millionen Einzelbildern, wovon etwa 100.000 intra-codiert sind und somit zur Suche herangezogen werden. Das Material ist eine Mischung aus Spielfilmen, Nachrichten- und Sportsendungen, sowie Zeichentrickfilmen.

Die Qualitätsmessung wurde in drei Kategorien für unterschiedliche Anwendungen unterteilt: Die Erkennung von Schlüsselbildern in laufenden Videos, das Retrieval von Videosequenzen mit Anfragen nach ganzen Bildern und zuletzt das Retrieval für die Anfrage mit Bildausschnitten.

6.1 Erkennung von Schlüsselbildern

Zwei Testszenarien für die Erkennung von Schlüsselbildern sollen im Folgenden beispielhaft vorgestellt werden: der Vorspann der Tagesschau mit eingeblendeter Uhrzeit und Informationstafeln zu Börsenkursen von n-tv, die bei entsprechenden Sendungen eingeblendet werden und eine Zusammenfassung der Informationen bieten. Beide sollen anhand der Ähnlichkeit zu einem vorgegebenen Vergleichsbild erkannt werden.

Der Tagesschauvorspann (Abbildung 2) stellt sich als der einfachere Fall heraus, da die Variationen geringer sind. Bei den n-tv-Börsentafeln sind größere Schwankungen im Hintergrund möglich. Die Tagesschau kann sowohl mit dem Farbmomente-Merkmal als auch mit dem direkten Vergleich der DC-Koeffizienten sicher erkannt werden.

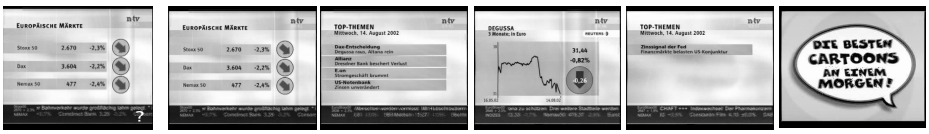

Abbildung 3: Die Erkennung der Informationstafeln ist wegen der großen möglichen Unterschiede schwierig. Die Chart-Tafel wurde nur mit dem Farbmomente-Merkmal erkannt, allerdings führt dies auch zu falschen Erkennungen, wie das letzte Bild zeigt.

Für die Erkennung der Börsentafeln (Abbildung 3) kann kein perfekter Schwellwert für beide Verfahren angegeben werden, jedoch liefert der Pixelvergleich bei einer festgelegten Grenze eine höhere Präzision, da dieses Verfahren für die Veränderung der durchschnittlichen Farbe nicht so anfällig ist. Auf der anderen Seite erkennt das Verfahren der Farbmomente Tafeln, die mit dem DC-Koeffizientenvergleich nicht erkannt werden.

Trotz dieser Schwierigkeiten ist eine sinnvolle Anwendung der Verfahren auch hier möglich. Ist der Benutzer beispielsweise daran interessiert, stündlich eine Zusammenfassung der Tafeln zu erhalten, so können diese mit einem niedrigen Schwellwert erkannt und gespeichert werden. Zwar würden auch einige nicht relevante Bilder gespeichert, der Benutzer kann diese jedoch schnell und problemlos aussortieren. Eine genauere Betrachtung der Börsentafeln liefert darüber hinaus einen Ansatz, die Erkennungsrate noch zu erhöhen, indem gesondert auf das untere Laufband und das Senderlogo getestet wird, so dass zumindest Fehltreffer innerhalb der Werbeblöcke ausgeschlossen werden können.

Zusammenfassend und mit Blick auf weitere Tests kann für die Erkennung von speziellen Schlüsselbildern festgestellt werden, dass diese mit den beiden einfachen Methoden relativ zuverlässig detektiert werden können, wenn sie unbewegt sind und sich voneinander kaum unterscheiden. Dabei müssen die Schwellwerte jedoch für den Einzelfall angepasst werden. Eine Kombination der beiden Verfahren führt zu einer weiteren Verbesserung der Erkennungsrate.

6.2 Retrieval von Videosequenzen mit Beispielframes

Zur Bewertung der Leistungsfähigkeit der vorgestellten Methoden zum Bildretrieval wurden Anfragen mit verschiedenen Beispielbildern gestellt und für eine Ergebnismenge der Größe 20 betrachtet. Ein Treffer steht dabei für eine ganze Videoszene und nicht nur für das Einzelbild, da ähnliche direkt benachbarte Bilder als ein Treffer zusammengefasst werden.

Über mehrere Versuche gemittelt ergibt sich, dass mit Farbmomenten 78% der Bilder relevant sind, beim Pixelvergleich sind es sogar bis zu 90%. Die Abbildungen 4 und 5 zeigen zwei Beispiele. Eine genauere Betrachtung zeigt, dass die Farbmomente eine größere Breite relevanter Bilder liefern, so dass sie sich insbesondere als erster Schritt von möglicherweise mehreren Anfrageschritten eignen, um dann mit einem besseren

Abbildung 4: Anfragebild zu einem Tagesschaubeitrag und Auszug aus der Ergebnismenge bei der Suche mit Farbmomenten. An Position 14 befindet sich als erster Nicht-Tagesschausprecher ein Nachrichtensprecher von n-tv. Der Treffer an Position 18 ist als Fehltreffer zu werten.

Abbildung 5: Suche nach einem Formel-1-Rennen durch Vergleich der DC-Koeffizienten. Die gezeigten Treffer 3 und 14 sind Fehltreffer.

gefundenen Bild eine zweite Anfrage mit dem Pixelvergleich zu starten.

Dies bestätigt sich auch in den durchgeführten Anfragen mit einer Skizze statt eines Beispielbildes. Die Farbmomente liefern hierbei im Mittel 47% relevante Bilder, der Pixelvergleich 35%. Ein Beispiel ist in Abbildung 6 zu sehen.

Sowohl bei der Anfrage mit einem Beispielbild wie auch mit einer Skizze existieren jedoch Einzelfälle, die den hier dargestellten Trends entgegenlaufen, so dass eine Berechnung beider Ähnlichkeitswerte und ihre Kombination viel versprechend erscheint, zumal der Rechenaufwand dadurch nur minimal ansteigt.

6.3 Retrieval von Videosequenzen mit Bildausschnitten

Deutlich schwieriger gestaltet sich die Suche, wenn nur ein Bildausschnitt gegeben ist, da der Informationsgehalt darin erheblich geringer ist. Zur Verbesserung der Ergebnisse wurde die Auflösung durch Berechnung von Subpixeln künstlich erhöht, sowie das Template in drei Skalierungs- und Rotationsstufen getestet. Damit wurde in diesen Testläufen eine Präzision von 60% in einer Ergebnismenge von 20 Bildern erzielt. In allen durchgeführten Tests wurde die Videoszene, aus der das Objekt ausgeschnitten wurde, mit einem hohen Rang in der Ergebnismenge zurückgeliefert, selbst wenn leichte Veränderungen in Größe oder Lage durchgeführt wurden. Ein Beispiel ist in Abbildung 7 gegeben.

7 Parallelisierung der Suche

Mit den in Abschnitt 4 beschriebenen Optimierungen können Laufzeiten erreicht werden, die für die Analyse einzelner Videos akzeptabel sind. Für eine größere Anzahl

Abbildung 6: Anfrage per Skizze. Die hier gezeigte Ergebnismenge wurde mittels Pixelvergleich berechnet. Der erste Treffer könnte als besseres Anfragebild für einen zweiten Suchlauf dienen. Nr. 5 und Nr. 20 sind Beispiele für falsch erkannte Szenen.

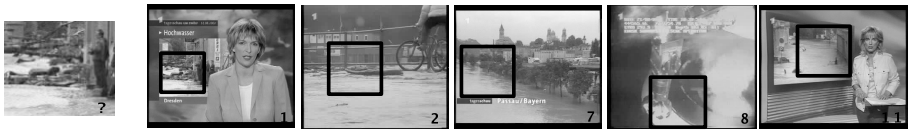

Abbildung 7: Die Anfrage zeigt ein Bild der Flut 2002 in Ostdeutschland, sowie einige ausgewählte Ergebnisse. Der erste falsche Treffer tritt für diese Anfrage an Position 8 in der Ergebnisliste auf.

von Videos, wie sie in Videodatenbanken vorliegt, ist jedoch der Einsatz von parallelen Konzepten unabdingbar.

7.1 Parallelisierung der Suchverfahren

Drei verschiedene Stufen der Parallelisierung sollen im Rahmen dieser Arbeit vorgestellt und untersucht werden:

Gleichzeitige Suche in verschiedenen Videos: Der einfachste Ansatz ist, für jedes zu durchsuchende Video einen eigenen Task zu starten. Gleichzeitig können dann genau so viele Tasks laufen, wie Prozessoren vorhanden sind (Abbildung 8a). Der Kommunikationsaufwand ist gering und beschränkt sich auf das Startsignal sowie das Zusammenfügen der Teilergebnisse. Nachteilig sind lange Leerlaufzeiten, die durch Videos unterschiedlicher Länge entstehen können und wenn die Anzahl der Videos kein Vielfaches der Anzahl der Prozessoren ist.

Gleichzeitige Analyse mehrerer Frames eines Videos: Bei diesem Ansatz werden mehrere Threads – einer pro vorhandenem Prozessor – zur parallelen Analyse eines einzelnen Videos erzeugt (Abbildung 8b). Nachdem das Video geladen wurde, werden die Video- von den Audiodaten getrennt (Demultiplexing). Dann beginnt die Analyse. Jeder Thread durchläuft das Video vom Anfang bis zum Ende. Trifft er auf ein I-Frame, das noch nicht analysiert wurde und sich auch derzeit nicht in Bearbeitung befindet, übernimmt dieser Thread die Bearbeitung und markiert es entsprechend in einer für alle Threads gemeinsamen Liste. Hierdurch ist zusätzliche Kommunikation erforderlich. Auf der anderen Seite werden Leerlaufzeiten auf die Zeit zur Analyse eines einzelnen Frames verkürzt. Nachteilig ist weiterhin, dass das Laden und Demultiplexen nicht parallel ausgeführt wird.

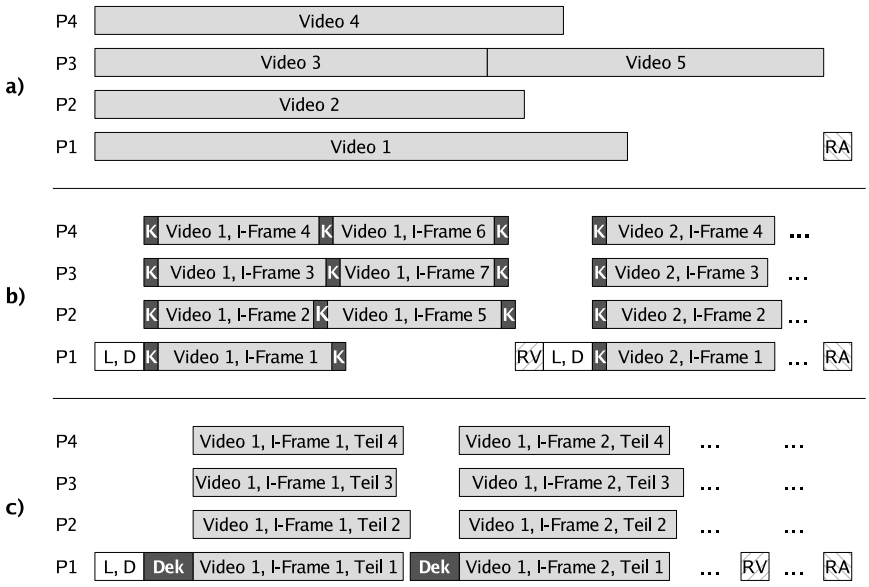

Abbildung 8: Gleichzeitige Suche in a) mehreren Videos, b) mehreren Frames eines Videos, c) verschiedenen Regionen eines Frames (L, D: Laden und Demultiplexen des Videos; K: Kommunikation und Suche nach dem nächsten freien Bild; Dek: Dekodierung eines Frames; RV: Zusammentragen der Teilergebnisse für ein Video; RA: Zusammenfügen des Gesamtergebnisses)

Parallele Analyse eines Frames: Jedes Bild wird in so viele Bereiche unterteilt, wie Prozessoren zur Verfügung stehen. Für jede Region wird ein Thread erzeugt, der in diesem Bereich die Suche durchführt (Abbildung 8c). Die Auslastung der Prozessoren ist bei dieser Methode sehr gleichmäßig. Nachteile sind ein erhöhter Kommunikationsaufwand und die sequentielle Ausführung bis zur Dekodierung des Bildes.

7.2 Parallele Architekturen

Zur Ausführung der gleichzeitigen Suche in mehreren Videos kommen verschiedene Parallelrechnerarchitekturen in Frage. Für große Bilddatenbanken mit mehr als vier Prozessoren wurde bereits die gute Eignung der Cluster-Architektur nachgewiesen [16, 10]. Der Vorteil liegt in der geringen Anzahl von Zugriffskonflikten beim Lesen aus dem Hauptspeicher und von der Festplatte im Vergleich zu SMP-Computern. Für kleinere Datenbanken ist kein signifikanter Unterschied erkennbar. Dieses Ergebnis kann mit geringen Unterschieden auf Videodatenbanken übertragen werden.

Um auf dem Cluster eine gute Auslastung zu erreichen, ist es zumeist notwendig, Videos temporär zwischen den Knoten zu verschieben, da bei einer mehrstufigen Anfrage Teilvideomengen ungünstig verteilt sein können. Angenommen, die gesamten Videos

sind gleichmäßig nach Erscheinungsdatum über die Knoten verteilt und der Benutzer möchte die dynamische Suche nur auf Videos eines bestimmten Regisseurs durchführen, so sind diese mit hoher Wahrscheinlichkeit nicht optimal verteilt.

Darüber hinaus ist das veränderte Verhältnis von Dateigröße zu Laufzeit zu beachten. Mit den vorgestellten Methoden kann eine hohe Datenmenge in relativ kurzer Zeit verarbeitet werden, da alle Informationen in den P- und B-Frames, die AC-Koeffizienten und insbesondere die Audioinformationen komplett überlesen werden. Hingegen dauert der Transfer mehrerer 100 MByte eines Videos deutlich länger als der von einzelnen Bildern. Hier liegt ein Vorteil der SMP-Architektur, wo alle Prozessoren den gleichen Zugriff auf die Videos haben.

Die Wahrscheinlichkeit für eine gute Verteilung der Videos steigt offensichtlich bei kleiner Anzahl der Knoten. Es stellt sich also die Frage nach einer guten Kombination aus SMP-Knoten, die zu einem Cluster zusammengefasst sind.

8 Laufzeiten und Speed up

In diesem Abschnitt sollen die unterschiedlichen Laufzeiten der Optimierungen für die sequentielle Suche sowie der Geschwindigkeitsgewinn durch die Parallelisierung betrachtet werden. Für die Zeitmessungen bei der sequentiellen Suche wurde ein derzeit handelsüblicher PC mit 1,8 GHz Pentium4-Prozessor zu Grunde gelegt. Für die parallele Suche kamen ein Vierprozessor-Alpha Rechner mit 600 MHz sowie ein Dualprozessor Xeon-System mit 2,2 GHz-Taktfrequenz – sowohl mit als auch ohne aktiviertem Hyperthreading – zum Einsatz. Die Hyperthreading-Technologie erlaubt den parallelen Einsatz mehrerer Funktionseinheiten der CPU, indem es dem Betriebssystem eine zweite CPU vortäuscht. Die Ergebnisse verschiedener Cluster-Architekturen wurden aus den zuvor experimentell gewonnenen Ergebnissen per Simulation und Extrapolation ermittelt.

8.1 Laufzeiten bei sequentieller Ausführung

Mit den nicht-parallelen Methoden kann die Suche nach Bildausschnitten in einem 875 Sekunden langen Video (193 MByte) in etwa 100 Sekunden durchgeführt werden. Bei dieser Zeit ist bereits eine Suche in drei Skalierungsstufen, drei Rotationsschritten und vier Subpixelpositionen im DC-Bild berücksichtigt.

Die Suche nach ganzen Bildern ist mit einer Bearbeitungszeit von 18 Sekunden erheblich schneller. Dabei entfallen ca. 6 Sekunden auf das Laden der Datei, das bei einer Schlüsselbilderkennung in Videostreams entfällt. Somit werden lediglich 2,1% bzw. 1,4% der Video-Laufzeit für die Analyse benötigt, so dass selbst mehrere Videostreams mit diesen Methoden problemlos gleichzeitig analysiert werden können und genügend Kapazität für verbesserte Vergleichsmethoden zur Verfügung steht.

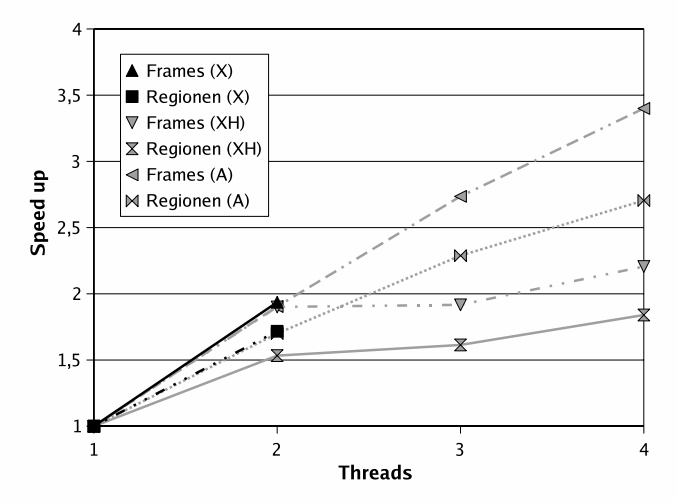

Abbildung 9: Parallelisierung der Suche in einem einzelnen Video. Getestet wurden ein Dualprozessor-Xeon-Rechner mit (XH) und ohne (X) Hyperthreading, sowie eine Alpha-Workstation mit vier Prozessoren (A).

8.2 Parallelisierung der Suche in einem einzelnen Video

Abbildung 9 zeigt den erzielten Speed up für die Suche in einem einzelnen Video. Zum Einsatz kommen hierbei das Verfahren zur gleichzeitigen Suche in mehreren Frames und zur gleichzeitigen Suche in verschiedenen Regionen eines Bildes.

Die höchste Beschleunigung wurde mit der Vierprozessormaschine erzielt und liegt bei 3,4. Aber auch der Dualrechner erreicht mit Hyperthreading einen Speed up von 2,2. Mit einer gezielten Ausnutzung der Eigenschaften der Hyperthreading Architektur kann dieser Wert vermutlich noch übertroffen werden.

Im Vergleich der Algorithmen ist festzustellen, dass die gleichzeitige Suche in mehreren Frames effizienter arbeitet als der feinere Parallelismus. Aus diesem Grund wurde davon abgesehen, weitere Ansätze zu testen, die die Parallelisierung auf noch tieferer Ebene einführen.

8.3 Parallelisierung der Suche in mehreren Videos

Im zweiten Testszenario soll die Analyse mehrerer Videos untersucht werden, wie es für den Anwendungsfall der Videodatenbanken zutrifft. Verglichen werden dabei die gleichzeitige Suche in verschiedenen Videos sowie die gleichzeitige Suche in verschiedenen Bildern desselben Videos. Die Ergebnisse sind in Abbildung 10 zu sehen. Die grobe Parallelisierung führt dabei zu einer höheren Beschleunigung und erreicht einen Maximalwert von 3,9 auf der Alpha-Workstation.

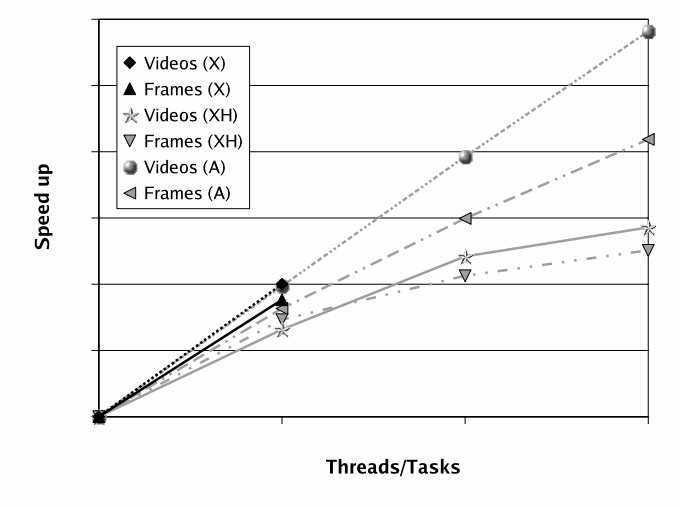

Abbildung 10: Parallelisierung der Suche in 100 Videos. Getestet wurden ein Dualprozessor-Xeon-Rechner mit (XH) und ohne (X) Hyperthreading, sowie eine Alpha-Workstation mit vier Prozessoren (A).

Bei diesem Test wurde jedoch angenommen, dass sich die Menge der Videos gleichmäßig über die Prozessoren aufteilen lässt. Dies ist jedoch – wie bereits in Abschnitt 7.2 gezeigt – nur selten möglich. Im folgenden Abschnitt wird daher ein realistischeres Szenario untersucht.

8.4 Parallele Suche in einem Cluster mit SMP-Knoten

Die in den letzten Abschnitten präsentierten Ergebnisse bilden die Grundlage für die Untersuchungen zum Aufbau eines möglichst leistungsfähigen Clusters. Die Daten sollen auf den Knoten so verteilt werden, dass die einzelnen Rechner jeweils disjunkte Teilmengen der gleichen Größe lokal speichern.

Im untersuchten Testszenario wird davon ausgegangen, dass der Benutzer zunächst eine Untermenge aller Videos auswählt, indem er beispielsweise den Regisseur oder ein Datum vorgibt. Die so gebildete Untermenge ist in der Regel nicht mehr gleichmäßig verteilt. Die Suche läuft dann nach der folgenden Strategie: Zunächst untersucht jeder Knoten die lokal gespeicherten Videos, wobei mit den größten Videos begonnen wird. Je nach Parallelisierungsstrategie werden dazu eine entsprechende Anzahl von Prozessen bzw. Threads erzeugt. Sobald ein Prozess auf einem Knoten kein neues Video mehr vorfindet, fordert es ein Video von einem anderen Knoten an und speichert es temporär zur Analyse auf seiner eigenen Festplatte, bevor es die Suche fortsetzt. Die Teilergebnisse müssen dann von einem Masterknoten zum Gesamtergebnisse zusammengetragen werden.

Videos	Knoten / CPU's	Xeon		Xeon (H)		Alpha	
		Parallelierung		Parallelierung		Parallelisierung	
		Videos	Frames	Videos	Frames	Videos	Frames
50	8 / 4	-	-	13,8	17,6	22,3	24,2
	16 / 2	22,8	28,1	18,9	25,8	22,7	27,1
	32 / 1	22,8	22,8	22,8	22,8	22,8	22,8
100	8 / 4	-	-	18,4	17,9	29,5	24,4
	16 / 2	30,4	29,4	25,2	27,1	30,0	28,2
	32 / 1	30,0	30,0	30,1	30,1	29,9	29,9
500	8 / 4	-	-	19,3	18,0	31,1	24,7
	16 / 2	31,8	30,0	26,4	27,6	31,6	29,0
	32 / 1	31,8	31,8	31,8	31,8	31,8	31,8

Tabelle 1: Erzielter Speed up bei der Analyse von 50, 100 und 500 Videos in unterschiedlichen Cluster-Konfigurationen mit insgesamt jeweils 32 Prozessoren. Der Xeon-Prozessor mit Hyperthreading wird dabei wie zwei Prozessoren gezählt. Die Knoten sind mit einem Hochleistungsnetzwerk wie etwa Myrinet verbunden. Die Werte sind durch Extrapolation und Simulation berechnet.

Getestet wurden unterschiedliche Cluster-Konfigurationen mit insgesamt jeweils 32 Prozessoren. Die in Tabelle 1 aufgeführten Ergebnisse zeigen, dass es nicht eine eindeutig beste Konfiguration für alle Anwendungsszenarien gibt. Ist die zu untersuchende Videomenge klein, so ist eine gleichzeitige Suche in verschiedenen Bildern eines Videos besser geeignet als eine gleichzeitige Suche in verschiedenen Videos, für große Videomengen ist das Umgekehrte der Fall. Die erste Aussage scheint zunächst den bisherigen Ergebnissen zu widersprechen. Die Ursache liegt jedoch nicht in dem Algorithmus selbst, sondern in der Tatsache, dass mit den kürzeren Ausführungszeiten pro Video eine bessere und gleichmäßigere Auslastung der CPU's erzielt werden kann.

Weiterhin zeigt sich, dass ein Einsatz von mehr als zwei Prozessoren pro Knoten für die kleine Videomenge nicht optimal ist, da der Algorithmus, wie in Abschitt 8.3 gezeigt, nicht besonders gut skaliert. Aber auch der Einsatz von 32 Einprozessorknoten führt nicht zur besten Beschleunigung, da sich 50 Videos nicht besonders gleichmäßig auf 32 Knoten verteilen lassen.

Bei größeren Datenmengen fällt eine nicht ganz gleichmäßige Verteilung hingegen nicht so stark ins Gewicht, da die gesamte Ausführungszeit im Vergleich dazu bereits sehr hoch ist. Hier überwiegt der Vorteil der besser skalierenden Methode. Es empfiehlt sich hier, eine oder zwei CPU's pro Knoten einzusetzen. Bei 500 Videos kann so ein nahezu optimaler Speed up von 31,8 erzielt werden.

Zusammengefasst bietet sich für die meisten Fälle eine Konfiguration an, die aus Knoten mit zwei Prozessoren besteht. Bei kleinen Datenmengen sollte die Suche parallel auf den Bildern eines Videos durchgeführt werden, bei größeren Datenmengen dagegen sollte entsprechend der Prozessoranzahl für jedes Video ein eigener sequentieller Suchprozess gestartet werden.

Abbildung 11: Zerlegung in einzelne Objekte beim Videoformat MPEG-4

9 Zusammenfassung und Ausblick

In dieser Arbeit wurden zwei effiziente Verfahren zum Bildvergleich innerhalb von MPEG-Videos vorgestellt, der Vergleich der DC-Koeffizienten, sowie der Farbmomente des DC-Bildes. Beide Methoden liefern hohe Erkennungsraten in kurzer Rechenzeit. Verbesserungen durch Kombination und Anpassung von Schwellwerten für spezielle Anwendungsfälle sind möglich, auch unter der Vorgabe der Echtzeitfähigkeit.

Neben der Suche nach kompletten Bildern wurde die Suche nach Bildregionen, Objekten oder Personen implementiert. Auch in diesem Fall wurde mit der einfachen Methode des DC-Koeffizientenvergleichs eine gute Trefferquote erreicht.

Zur Bewältigung des großen Rechenaufwands wurden verschiedene parallele Strategien entwickelt und vergleichend untersucht. Dabei zeigte sich, dass ein grober Parallelismus einem feinen in den meisten Situationen überlegen ist. Ein Vergleich verschiedener Parallelrechnerarchitekturen hatte zum Ergebnis, dass ein Cluster aufgebaut aus Knoten mit zwei CPU's bei großen Datenmengen einen nahezu optimalen Speed up ermöglicht.

Zukünftige Arbeiten sollen Kamera- und Objektbewegungen berücksichtigen. Ferner steht die Entwicklung ähnlicher Verfahren für das moderne Videoformat MPEG-4 im Mittelpunkt des Interesses, insbesondere vor dem Hintergrund, dass im MPEG-4-Format Objektstrukturen gespeichert werden können, wie in Abbildung 11 dargestellt.

Literatur

[1] Informationen über das DVB-Projekt, Standards und Whitepapers sind im WWW unter der URL www.dvb.org zu finden.

[2] ISO/IEC 11172-1 (-2, -3, -4, -5), Information technology — Coding of moving pictures and associated audio for digital storage media – Part 1 (2, 3, 4, 5): Systems (Video, Audio, Compliance Testing, Software Simulation), 1993.

[3] ISO/IEC 13818-1 (-2, -3, -4, -5, -6, -7, -9, -10), Information technology – Generic coding of moving pictures and associated audio information – Part 1 (2, 3, 4, 5, 6, 7, 9, 10): Systems (Video, Audio, Conformance testing, Software simulation, Extensions for DSM-CC, Advanced Audio Coding (AAC), Extension for real time interface for systems decoders,

Conformance extensions for Digital Storage Media Command and Control (DSM-CC)), 1998-2000.

[4] ISO/IEC 14496-1 (-2, -3, -4, -5, -6, -7) Information technology – Coding of audio-visual objects – Part 1 (2, 3, 4, 5, 6, 7): Systems (Visual, Audio, Conformance testing, Reference software, Delivery Multimedia Integration Framework (DMIF), Optimized reference software for coding of audio-visual objects), 2001-2003.

[5] Whitepapers zu Windows Media stellt Microsoft unter der folgenden Adresse bereit: http://www.microsoft.com/windows/windowsmedia/technologies/overview.aspx

[6] Whitepapers von RealNetworks sind unter der URL http://www.realnetworks.com/industries/resources/literature/index.html abrufbar.

[7] S.-F. Chang, W. Chen, H. Meng, H. Sundaram, D. Zhong, VideoQ: An Automated Content Based Video Search System Using Visual Cues. *Proc. of ACM Multimedia*, pp. 313–324, 1997.

[8] A. Hampapur, A. Gupta, B. Horowitz, C.F. Shu, C. Fuller, J. Bach, M. Gorkani, R. Jain, Virage video engine. *Proc. SPIE: Storage and Retrieval for Image and Video Databases*, pp. 188-197, 1997.

[9] D. Ponceleon, S. Srinivasan, A. Amir, D. Petkovic, Key to Effective Video Retrieval: Effective Cataloging and Browsing. *Proc. of ACM Multimedia*, pp. 99–107, 1998.

[10] O. Kao, Dynamisches Retrieval von multimedialen Daten auf parallelen Architekturen. *Habilitation an der TU Clausthal, Shaker Verlag*, 2002.

[11] O. Kao, S. Stapel, Case study: Cairo – a distributed image retrieval system for cluster architectures. In T.K. Shih, editor, *Distributed Multimedia Databases: Techniques and Applications*, pp. 291–303. Idea Group Publishing, 2001.

[12] M. A. Stricker, M. Orengo, Similarity of Color Images. Storage and Retrieval for Image and Video Databases (SPIE), pp. 381–392, 1995.

[13] R.-F. Chang, W.-J. Kuo, H.-C. Tsai, Image retrieval on uncompressed and compressed domains. In *Proceedings of International Conference on Image Processing*, pp. II-546-549, 2000.

[14] G. Falkemeier, G.R. Joubert, O. Kao, A system for Analysis and Presentation of MPEG Compressed Newsfeeds. In *Business and Work in the Information Society: New Technologies and Applications*, IOS Press, pp. 454-460, 1999.

[15] K. Shen, E. Delp, A fast Algorithm for Video parsing using MPEG compressed Sequences, *Proc. of the IEEE International Conference on Image Processing*, pp. 252-255, 1995.

[16] T. Bretschneider, S. Geisler, O. Kao, Simulation-based Assessment of Parallel Architectures for Image Databases, *Proc. of the International Conference on Parallel Computingg (ParCo 2001)*, Imperial College Press, pp. 401-408, 2001.